NDN DeLorean: An Authentication System for Data Archives in Named Data Networking

Yingdi Yu
UCLA
yingdi@cs.ucla.edu

Alexander Afanasyev
Florida International University
aa@cs.fiu.edu

Jan Seedorf
HFT Stuttgart
jan.seedorf@hft-stuttgart.de

Zhiyi Zhang
UCLA
zhiyi@cs.ucla.edu

Lixia Zhang
UCLA
lixia@cs.ucla.edu

ABSTRACT

Named Data Networking (NDN) enables data-centric security in network communication by mandating digital signatures on network-layer data packets. Since the lifetime of some data can extend to many years, they outlive the lifetime of their signatures. This paper introduces *NDN DeLorean*, an authentication framework to ensure the long-term authenticity of long-lived data. The design of DeLorean takes a publicly auditable bookkeeping service approach to keep permanent proofs of data signatures and the times when the signatures were generated. To assess DeLorean's feasibility, the paper presents a set of analytical evaluations on the operational cost as a function of data archive volumes. The paper also identifies several remaining issues that must be addressed in order to make DeLorean a general solution to authenticating long-lived data.

CCS CONCEPTS

• **Security and privacy → Security services**; *Security protocols*;

KEYWORDS

NDN, Authenticity, Signature Logger

ACM Reference format:
Yingdi Yu, Alexander Afanasyev, Jan Seedorf, Zhiyi Zhang, and Lixia Zhang. 2017. NDN DeLorean: An Authentication System for Data Archives in Named Data Networking. In *Proceedings of ICN '17, Berlin, Germany, September 26–28, 2017,* 11 pages.
DOI: 10.1145/3125719.3125724

1 INTRODUCTION

Named Data Networking (NDN) changes the network communication model from "delivering packets to an end host" to "retrieving (immutable) data by name." This change enables many of the long sought-after network properties, including efficient data distribution via multicast and in-network storage, ad hoc and delay-tolerant communication, and many more.

One of the key NDN mechanisms is building security (authenticity) into the network-level data packets—each data packet[1] must carry a digital signature to ensure its authenticity, whether it is being transmitted or stored. However, digital signatures generally have a limited lifespan, e.g., the current practices set validity for signatures of PKI certificates from several months to several years [2]. This limited lifetime works well for today's session-based network security model, where certificates (their signatures) only need to be valid at the time of establishing a session. With NDN's data-centric security model, on the other hand, the lifespan of signatures can be much shorter than that of some long-lived data they authenticate. This mismatch between the signature lifetime and the data lifetime poses a serious challenge. A signature that used to successfully authenticate a data packet a few years ago may no longer be trusted today, while the data itself may still be relevant, especially in cases of historical data archives.

In this paper, we propose an authentication framework for NDN data archives, dubbed NDN DeLorean, which uses a *look back* data authentication model: long-lived data can be authenticated by checking the signature validity at the time of the data creation. Inspired by the Certificate Transparency [10], DeLorean provides a publicly auditable bookkeeping service that issues proofs of data packet signature existence at a given time by logging the fingerprints of the signatures in the form of a Merkle tree. Given a data packet with its signature, the certificates that authenticate the signature, and the DeLorean proofs for the signature of the data packet and certificates, one can perform the authentication regardless of the signatures expiration.

Our main contributions in this work include: (1) the use of look back validation model as the solution to long-lived data maintenance in NDN (Section 4), (2) the design of the first publicly auditable signature bookkeeping service over NDN and its initial implementation prototype [25] (Section 5), and (3) an analytical evaluation of the scalability of our proposed solution (Section 6).

We believe that the DeLorean framework is an important step toward effective authentication of long-lived data. We also identify a number of remaining issues (Section 7) that must be addressed to make DeLorean a general solution, in particular the challenge of scaling the design to handle unpredictable data volumes in the future.

[1]Or a group of packets, when a manifest-like approach is used [3, 14]

2 BACKGROUND

Named Data Networking (NDN) is a proposed networking architecture that uses data fetching as a communication primitive. NDN defines two types of network packets to support the data-centric communication: *interest packets* as the request to retrieve desired data, and *data packets* that carry the actual data. Data consumers send interests and NDN routers forward interests based on the names carried in the interests toward potential data sources, setting up a state of "pending interests" along the way. Upon receiving an interest, a data producer returns the corresponding data packet which may already exist or is created on demand. NDN mandates that the producer must secure each data packet as soon as it is created, using cryptographic signature to bind together the name and content of the packet. This data packet is returned back to the consumer or consumers following the breadcrumb path of pending interest states. Because each data packet is immutable and identified by a unique name, and can be authenticated on its own, NDN routers can cache data packets to satisfy subsequent interests for the same data.

For illustrative purposes, in the rest of the paper, we use an electronic version of "USA Today" newspaper as an example. We assume that USA Today publishes all its articles under a namespace with the name pattern "`/UsaToday/[Date]/[Category]/[ArticleID]`". For example, a data packet with the headline story "Youth Jailed" from October 22, 2015 would be named "`/UsaToday/2015/10/22/headline/YouthJailed`".[2] All the data packets of an article are signed by the keys of its author, who should have a unique name under the namespace "`/UsaToday/journalist`". For example, Compu Fax, the author of "Youth Jailed" article,[3] would sign the above article with the key "`/UsaToday/journalist/CompuFax/KEY/_v=10`". Note that NDN eliminates the requirement that data producers (USA Today) and consumers (its readers) have to be online at the same time to communicate. A data producer can move data packets to an external data repository (repo) to serve future requests for the data, or the data can be carried by some devices to meet future consumers as in a DTN scenario.

2.1 Data-Centric Authenticity

Since the data-centric communication model enables consumers' interests "to pick data from anywhere possible", traditional session-based security solutions, e.g., TLS [7], IPSec [9], are no longer applicable as they are designed to create secure connection between two fixed end nodes. Although recent work (e.g., CoAP based on DTLS [19]) eliminates the concept of connection at the network layer, the concept of secure session remains in the application layer. NDN embraces a complete *data-centric security* which ensures integrity, provenance, and secrecy of data itself, instead of relying on the security of delivery session.

NDN data producers attach digital signatures to data packets and consumers who have the producer's public key can directly authenticate such data, applying the trust schema [24] (i.e., that names of data and keys are as expected and certification chain terminates in a pre-trusted key—*trust anchor*), not worrying about from where this data came. If consumers do not have the producer's public key at hand, they can retrieve the key (plus the so called *certification chain*, including the signing key of the producer's key, the signing key of the signing key, etc.) following the information in the "`KeyLocator`" field of data packets and recursively applying the trust schema. Figure 1 shows an example of certification chain for our USA Today example. Note that keys are just another type of general NDN data, and given the data packet carrying a public key binds the key to its name through the packet's signature, it is effectively a public key certificate.

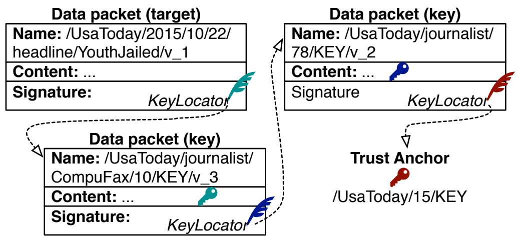

Figure 1: An example of certification chain consisting of target data, intermediate keys, and trust anchor

2.1.1 *Limited Cryptographic Signature Lifetime.*

Digital signatures usually have a limited lifetime: signature algorithm can get broken, private keys can get compromised or even reconstructed, provided enough time and computation power. This limit is generally captured as part of a data packet signature in the form of the validity period field, which defines a period when the signature is considered valid. Moreover, the *effective* validity period of a signature can be shorter than its defined validity period, as it is the intersection of the validity periods of all the keys along the certification chain. For the USA Today article in our example, if the article's signature was set to be valid for ten years, while CompuFax key's signature was only valid for one year, then the effective time when the article's signature would be verifiable is only that one year. Therefore, to be able to verify the validity of data packets over prolonged time periods, one has to either (1) set the validity periods of all signatures to unrealistically long values, (2) keep refreshing signatures of the data and keys along the certification chain, or (3) use a secure mechanism to verify signatures considering the validity at the time signatures were created, e.g., NDN DeLorean proposed in this paper.

2.1.2 *Authentication Granularity.*

Public key cryptography is computationally expensive, which makes it impractical with currently available technologies to create separate public key signatures for individual data packets. However, it is possible to mitigate these limitation by using only one public key signature per group of data packets [3, 14, 20], securing the whole group using a relatively cheap symmetric cryptography and/or cryptographic hashing. The size of the packet group, in this case, will determine the granularity of the authentication: individual packets can be authenticated only in the context of the whole group.

2.2 Merkle Tree

Merkle tree [15] is a special k-ary tree data structure, where the value of each node is the hash of the concatenation of the values of

[2]A large article can be split into multiple data packets, each getting a unique name with this prefix and a suffix that represents the segment number, e.g., "`_s=1`", "`_s=2`", etc.

[3]Compu Fax is indeed a program that can write automated stories for USA Today in "Back to the Future" film.

its children. Similar to hash chains [17] in which the last node seals all the previous nodes so that they cannot be altered (unless the hash algorithm is broken), the root node of the Merkle tree seals all the leaves in the tree. Any change of any leaf leads to the change of the corresponding intermediate nodes and ultimately the root hash.

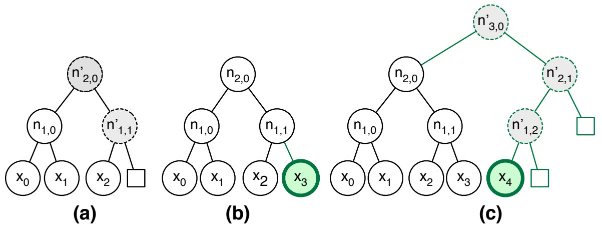

Figure 2: Merkle tree evolution

Figure 2 demonstrates the construction of a binary Merkle tree, initially with only three leaf nodes. When adding a new leaf node to the tree, there are several cases of how the tree is updated. If there is an unused position at the bottom level (level zero) of the tree, the node will simply take that position (transition Figure 2a ⟶ Figure 2b when x_3 is added). If there are no unused positions and the tree is complete, then the tree will need to grow by an additional level and the added node will take the newly created unused position (transition Figure 2b ⟶ Figure 2c when x_4 is added). If the tree is not complete, then a new intermediate level and new unused positions for leaf nodes will be created. In the first and third case, the addition changes the hash value of the nodes in the subtrees corresponding to the added node ($n_{1,1}$ recalculated using x_2 and newly added x_3, $n_{2,0}$ recalculated using $n_{1,0}$ and newly calculated $n_{1,1}$). In the second case, the tree gets a new root node, calculated using the previous root and the newly inserted branch of the tree.

Note that until a level of the Merkle tree is complete, the values of the parent nodes on the path from the incomplete branch to the root are not stable and will change whenever a new leaf node is added.

3 THREAT MODEL AND ASSUMPTIONS

In this paper, we focus on the authenticity of low-volume *long-lived* public data, i.e., a relatively small set of data packets or data collections that need to be preserved for a long period of time. Typical examples of such data include newspaper articles, library archives, historical records, experimental results, etc. We focus specifically on ensuring that the authenticity of these long-lived data can stay verifiable, potentially many years after the data producer ceased to exist. We assume that the consumers know which trust schema should be used to authenticate data and that the trust anchors do not change over time (i.e., consumers know how to construct the valid certification chains). We also assume that the cryptographic keys in the corresponding certification chains of the long-lived data have limited validity periods and the keys are not compromised/leaked during their validity periods, i.e., there are no *producer impersonations* while the certification chains are within their validity period. In our future work, we plan to investigate the generic applicability of DeLorean to other types of higher volume data, how to ensure the long-term secrecy of confidential data, and how to mitigate keys' compromise and revocation during their validity periods.

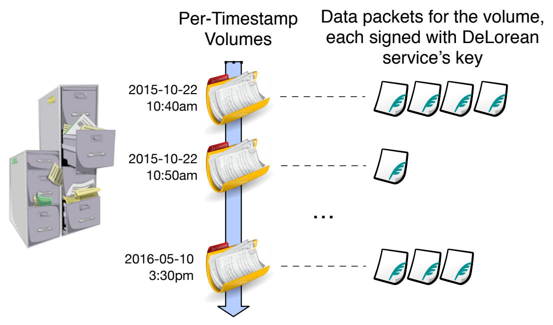

Figure 3: DeLorean's data chronicle

In this work, we assume that attackers can reconstruct the signing key after the validity period—through key leaking or brute force computation. In this case, an attacker may impersonate the key owner by generating a signature on fake data using the reconstructed key. Our counter solution to this is to provide a bookkeeping service that "certifies" the existence of data at particular points of time. We do not assume that the bookkeeping service is trustworthy and never misbehaves. We consider the following four potential misbehaviors of the bookkeeping service: (1) *denial* of providing the previously issued proofs, (2) *repudiation* (pretending that the previously issued proofs are invalid), (3) *reordering* (altering the time relation of the existing proofs), and (4) *injection* (injecting a new proof in the past). To ensure the correctness of the bookkeeping service, we leverage public auditing, i.e., a large group of auditors occasionally validate the bookkeeping history to defer the service from misbehaving. We assume that the majority of auditors perform the validation on time and report misbehavior immediately when it happens.

In the following Sections, we describe the design of DeLorean framework and how it prevents the above misbehaviors through constant *public auditing* of the historical record of the issued proofs.

4 DELOREAN DESIGN OVERVIEW

DeLorean is an always-on service that publishes a data "*chronicle*" (Figure 3). The chronicle consists of a sequence of *volumes*, each containing fingerprints of the witnessed signatures (e.g., signature of an individual data packet or signature of a manifest) within a fixed timeslot. During the timeslot, DeLorean accumulates in the "current" volume signatures from publishers who want their signatures to be recorded, finalizing and publishing the volume as a set of NDN data packets at the end of the timeslot. As explained in Section 5.6, after the volume is finalized, it cannot be changed without invalidating consistency with any future volumes, which can be easily detected by auditors. The presence of a signature in a particular volume is effectively a *timestamp proof* that the signature existed before the end of the corresponding time slot. For example, DeLoran chronicle could have witnessed a signature of the data packet for "`/UsaToday/2015/10/22/headline/YouthJailed`" USA Today article as part of 10:50am volume on October 10, 2015. Until the time travel becomes a reality and for as long as the data chronicle is consistent and the corresponding volume can be retrieved, one would be able to authenticate this article in perpetuity.

While the signature is still valid, a data producer (article's author) or an archive service on the producer's behalf (USA Today

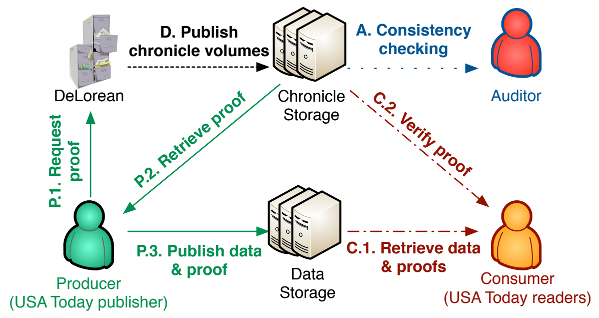
D. Publish chronicle volumes
A. Consistency checking
DeLorean
Chronicle Storage
Auditor
P.1. Request proof
P.2. Retrieve proof
C.2. Verify proof
P.3. Publish data & proof
C.1. Retrieve data & proofs
Producer (USA Today publisher)
Data Storage
Consumer (USA Today readers)

Figure 4: DeLorean workflow

publisher) can request a timestamp proof for this signature from DeLorean (Flow P.1 in Figure 4), supplying its fingerprint in the form of a hash digest. The response to this request is a name of the chronicle volume that will be published by DeLorean at the end of the current cycle (Flow D) and the index of the fingerprint in the volume. After waiting until the volume is ready (on average a 5-minute wait in our example), the producer can retrieve the volume to verify whether DeLorean has included the fingerprint in the volume (Flow P.2). In the end, the producer can publish the timestamp proof of the signature—the full name of the volume and the index of data fingerprint—alongside the data (Flows P.3).

To verify data despite its signature's expiration, consumers need to "look back" to the timepoint when the data's signature was witnessed by DeLorean. To securely do so, a consumer needs to obtain the corresponding timestamp proof signature (Flow C.1) and verify this proof with the DeLorean service, if necessary fetching additional chronicle volumes (see Section 5). Similarly, the consumer can verify signatures of the signing keys, signing keys of the signing keys, etc. (i.e., the trust chain) With all timestamp proofs, the consumer can verify the authenticity of data signature, as if it was at the time of time-stamping.

In order to ensure the correct and truthful operations of DeLorean, a number of auditors must continuously check the consistency of the chronicle (Flow A), i.e., checking that DeLorean has not modified the previously published volumes. The more auditors are involved in the process, the less frequently each individual auditor needs to perform consistency checking (Section 6.3). If an auditor detects that DeLorean has modified the chronicle through obtaining several mutually inconsistent volume records, represented as NDN data packets signed by the DeLorean provider [4], it can share them publicly as discussed in Section 7.4. This public-eye audit serves as a deterrent from the service's misbehavior, in the worst case requiring a transition to another provider preserving the consistent part of the chronicle.

4.1 Design Discussion

The look-back validation service realized by the NDN DeLorean service essentially requires "certification" of the timestamp while the signature is still considered valid, e.g., just after it is created. Such certification can be implemented in several different ways.

[4]As a first design, we focus on a single provider that maintains the chronicle and volumes, concentrating on the design of an NDN naming scheme for DeLorean and a scalability analysis of such a system. Future work may investigate more decentralized ways of maintaining chronicle and volumes (e.g. as in bitcoin).

One approach is a dedicated trusted service (e.g., Apple's timestamp certification service) which issues the signed timestamps. To ensure the long-term timestamp validity, the service would need to periodically update its signed keys and even re-issue timestamps. While this approach provides relative strong non-repudiation guarantees, the drawback is a "blind" trust that the service behaves correctly, without the ability to reliably detect out-of-order or back-issued timestamps.

Another approach is to use a block-chain (hash-chain) approach, where each new signature is added to the chain and potentially associated with a timestamp; while the chain is distributed across many (all) interested parties. In this way, the existence of a signature in the chain indicates its validity at the time (either absolute time or relative to other signatures); and signatures cannot be added or removed in the middle of the chain without breaking chain's consistency.

NDN DeLorean belongs to the second approach and realizes the concept of a hash-chain efficient form in terms of storage and verification time. In DeLoran chronicles, each item in the "chain" represents an aggregation of all signatures issued within a specific time range and "chain" (the chronicle) is organized as a Merkle tree allowing fast $O(\log n)$ lookup to verify consistency.

5 DELOREAN

The design of DeLorean, largely inspired by the Certificate Transparency framework [10] [11], realizes the look-back signature validation via creating proofs that certify the existence of the signatures at the specific times in the past. These proofs are recorded in the DeLorean *Chronicle* tree and grouped into per time period *Volume* trees. Similar services exist on the Internet today [13]. Our contribution is the design of such a system and corresponding naming scheme for NDN as well as a scalability analysis of this approach. Further, we propose and analyze a decentralized, public audit of the DeLorean service provider and show how NDN can facilitate this public audit. In the following, we describe how Chronicle and Volumes trees are constructed and published in the NDN network, what are the mechanisms and algorithmic complexity to verify existence of a signature in the chronicle, as well as show details of the publicity-based trust mechanisms that underlie the security properties of DeLorean.

5.1 Chronicle Construction

The DeLorean chronicle is a multi-dimensional Merkle tree (Figure 5), whose overall state is reliably captured by the hash of the root node (chronicle digest). The chronicle digest state is calculated using the top-level Merkle structure ("Chronicle tree") that records causal relationships between volume states (volume digests), i.e., that the volume v_4 goes after the volumes v_2 and v_3, etc. The volume digests are calculated using their own Merkle trees ("Volume trees") that capture signatures (their cryptographic hashes) witnessed during the time interval corresponding to the volume.

At the end of each time interval, the DeLorean service adds a new volume digest node to the Chronicle tree. To do so, the service creates a set of data packets corresponding to the newly inserted and updated nodes, signing each data packet with the service's private key. In other words, the signature certifies that the chronicle digest

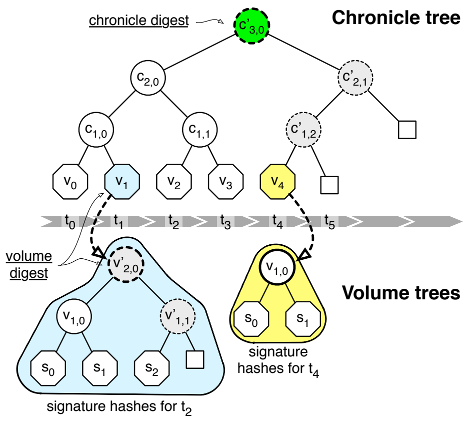

Figure 5: Two-level hierarchy of the timestamp service

(and all other children digests) was produced by the DeLorean service; while the digests provide assurances of the Chronicle and Volume trees consistency.

5.2 Volume Construction

Each DeLorean volume is a set of signatures (their cryptographic hashes) that were witnessed during the corresponding time period. During the current time period, the DeLorean service collects the signatures from the publishers by adding them as leaves to the volume tree. As soon as the time period is over, the DeLorean adds the latest version of the volume digest to the chronicle tree, effectively "sealing" the volume from further modifications. After this, the publishers can contact the service and retrieve DeLorean proofs of signature existence during the volume's time period, which can be used to reliably roll-back clocks for future validation of their data.

5.3 Proof of Existence

In order to prove the existence of a particular leaf node in a Merkle tree, one needs to be able to reconstruct a part of the tree along the path from the leaf to the root. For example, given the current state of the chronicle represented by its digest $c'_{3,0}$, to check that the volume with digest v_1 exists in the Chronicle tree (Figure 5), the path $v_1 \rightarrow c_{1,0} \rightarrow c_{2,0} \rightarrow c'_{3,0}$ needs to be reconstructed: $c^?_{1,0} = \text{hash}(v_0, v_1)$, $c^?_{2,0} = \text{hash}(c^?_{1,0}, c_{1,1})$, $c^?_{3,0} = \text{hash}(c^?_{2,0}, c'_{2,1})$. After that, equality between $c^?_{3,0}$ and $c'_{3,0}$ serves as a proof of chronicle consistency and that v_1 is part of it. Similarly, one can prove that signature s_1 exists in the Volume tree using the equality between reconstructed volume tree root $v^?_{2,0}$ and a known volume digest v_1.

The computational cost to prove the existence of a single volume is $O(\log V)$ hash computations, where V is the total number of volumes in the chronicle; and the computational cost to verify the existence of a signature in the chronicle is $O(\log V + \log S)$, where S is the (average) number of signatures witnessed per time period. The arity of the Merkle tree used for Chronicle and Volume trees will determine the base of the logarithm in the above equations.

Given a volume tree, a consumer can quickly locate a record according to the local record index. Based on the local index, a consumer can compute a verification path from the record back to the volume tree root and verify the existence of the record in a volume, same way as verifying the volume existence in chronicle.

In summary, to assure that a specific signature was witnessed by the DeLorean service at a specific time interval, the consumer needs to have: (1) volume digest, (2) volume index, and (3) index of the signature within the volume. The first two are used to reconstruct the relevant portions of the chronicle tree; and the last one along with the fingerprint of the data signature (obtained from data directly) can reconstruct and verify consistency of the volume tree.

5.4 Publishing Node States

In order to successfully prove the existence of any volume or signature in the chronicle, one needs to obtain sibling nodes in the chronicle and volume trees. To allow that, DeLorean leverages the power of NDN to publish each such node as an immutable, named, and cryptographically signed NDN data packet. We use the following naming convention for these data packets (Figure 7): "`/<prefix>/<TREE-TYPE>/<NODE-STATE>/<NODE-INDEX>/<DIGEST-VALUE>`", where

- "`<prefix>`" is a prefix to identify instance of the DeLorean chronicle service, e.g., if there is only one service in the world, this could be just "`/DeLorean`";
- "`<TREE-TYPE>`" is an identification of the tree type and can be either "`_CHRONICLE`" or "`_VOLUME-<ID>`" for the chronicle or a specific volume tree. "`<ID>`" in the volume tree name is a sequence number of the volume, since the beginning of DeLorean service instance;
- "`<NODE-STATE>`" is the state of the Merkle tree node, which is either "`complete`" when a node has the full set of descendents or "`incomplete=<ID>`" when one or more descendents do not yet exist. Recall that until a subtree of the Merkle tree is complete, the root digest of this tree changes whenever a new leaf node is added. "`<ID>`" in the incomplete tree is used to disambiguate the name for different incomplete states.

  The naming of the incomplete state nodes is designed to simplify retrieval of the latest state, is consistent across all incomplete nodes at a given time, and is directly related to the number of nodes in the tree. In other words, at any given point of time, the current state of any node in the chronicle tree is determined by the current number of volumes. For a 32-ary chronicle tree with the largest volume sequence number s, for the node with index i at the level l (both i and l start from 0) :

$$\text{“}\texttt{<NODE-STATE>}\text{”} = \begin{cases} \text{“}\texttt{complete}\text{”}, & \text{if } s \geq 32^l \times (i+1) \\ \text{“}\texttt{incomplete-(s+1)}\text{”}, & \text{otherwise} \end{cases}$$

  For example in Figure 6, all incomplete nodes are published using the "`incomplete-2050`" state.
- "`<NODE-INDEX>`" is a tuple of the level of the node in the Merkle tree (0 for leaf nodes, up to $\lceil \log_k N \rceil$ for parent nodes, where N is the number of leaf nodes and k arity of the Merkle tree) and the index of the node at the specified level (from 0 to k), such as "`2,1`".

  Given the sequence number s of the leaf node of interest, and the desired level l of the intermediate node, its index is $(l, \lfloor s \times k^{-l} \rfloor)$.

- "`<DIGEST-VALUE>`" is the digest value of the Merkle tree node.

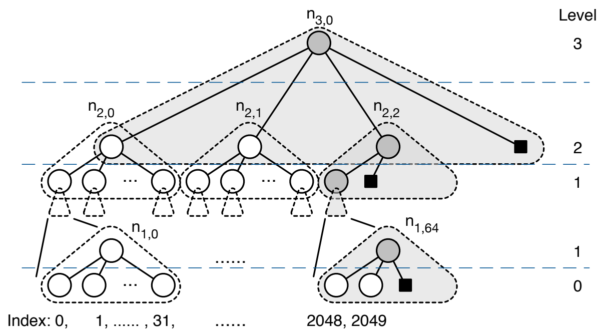

Figure 6: 32-ary Merkle tree example

Each published node data packet includes the hash values of all its children. With a 1500-byte MTU and SHA-256 hash algorithm, a single data packet can safely carry at least 32 SHA-256 hashes (1024 bytes in total), leaving enough space for other fields in the data packet. For this reason, we chose a 32-ary Merkle tree (exemplified in Figure 6) to construct DeLorean's chronicle and volumes and publish data packets for each node of the trees (Figure 7). Note that the leaf nodes of the Chronicle tree are published only as root nodes of the corresponding Volume trees and their hash is simply included in the data packets that represent the first-level of the Chronicle tree.

Each data packet representing a node in the tree is signed by the DeLorean provider and can be independently authenticated within its signature validity [5]. At the same time, to authenticate the whole chronicle it is enough to authenticate just one data packet representing the root node: the rest of the tree can be implicitly validated using the included cryptographic hashes. Therefore, the Chronicle and Volume trees can be successfully authenticated as long as the signature of the root node is valid, which is true as long as the service keep witnessing and recording signatures.

Note that retrieval of tree nodes leverages efficient data distribution of NDN: requests from multiple requesters can be efficiently joined or served from in-network caches. Because nodes at higher layers are involved in more verifications, they are more frequently requested and have a high chance of being universally cached in the network.

Also note that all nodes in all states are represented as immutable data packets, and can be easily replicated in the network. At the same time, as we describe in the next session, only nodes with complete state and the latest versions of the nodes in incomplete state need to be preserved by the DeLorean service provider and auditors.

5.5 Incomplete Chronicle Node States

The chronicle digest and rightmost intermediate nodes at each level of the Chronicle tree will be represented as incomplete state nodes almost all the time, only occasionally becoming complete (i.e., only just before the tree needs to grow one level up). For example, an incomplete node $c'_{3,0}$ in Figure 5 captures the state of volumes $v_0, \ldots v_4$, which will change as soon as a new volume added to the chronicle.

[5]NDN supports short-lived signatures as an alternative for key revocation. Signatures from the DeLorean service, however, may have a rather long lifetime, as the keys used for signatures are assumed to be reasonably protected. For such keys thus key revocation mechanisms may be considered.

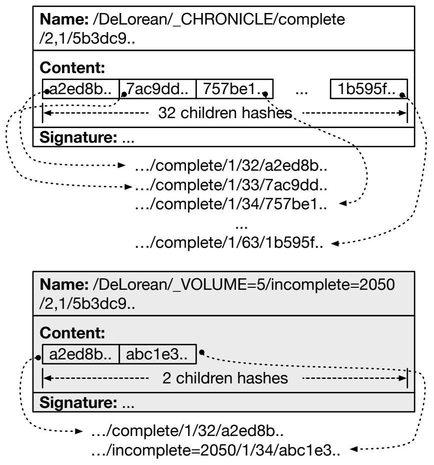
Name: /DeLorean/_CHRONICLE/complete /2,1/5b3dc9..
Content:
a2ed8b..
7ac9dd..
757be1..
...
1b595f..
32 children hashes
Signature: ...
.../complete/1/32/a2ed8b..
.../complete/1/33/7ac9dd..
.../complete/1/34/757be1..
...
.../complete/1/63/1b595f..
Name: /DeLorean/_VOLUME=5/incomplete=2050 /2,1/5b3dc9..
Content:
a2ed8b..
abc1e3..
2 children hashes
Signature: ...
.../complete/1/32/a2ed8b..
.../incomplete=2050/1/34/abc1e3..

Figure 7: Examples of 32-ary chronicle and volume tree node data packets

This fact, however, does not impact the ability to perform the tree reconstruction nor requires keeping data packets that represent all versions of the incomplete states. Because each newly published version of the incomplete node includes all previously existing child node digests, it is effectively a superset node and only the latest version would be needed to perform any required tree reconstruction. Therefore, as soon as the new incomplete node is created, the data packet representing the old incomplete version of the same node can be safely removed from the system.

5.6 Publicity-Based Trust / Public Audit

One of the objectives behind the design of DeLorean is to avoid dependency on a complete trust of a single operational party. While DeLorean still relies on a single party to create and publish Chronicle and Volume trees based on witnessed signatures, the overall trust of the service comes from the consistency verification by public eyes. More specifically, a DeLorean service provider is trusted only while its behavior is consistent, as monitored by "public auditors" (e.g., librarians who are dedicated to ensuring consistency of newspaper archives) or dedicated set of trust auditors (e.g., provided by Google, Amazon, Microsoft, etc.). Moreover, to ensure trustworthiness of the time periods of volumes, the state of each newly published volume needs to be checked at the time it is published. Since the producers of data recorded by DeLorean rely on it to provide the existence proofs, they have a strong motivation to audit the consistence as soon as volume information becomes available. With the trivial overhead of DeLorean consistency verification, as shown in Section 6, this can be easily achieved.

A DeLorean service, once being detected as inconsistent (i.e., a previous volume being modified), immediately loses its trustworthiness. We expect that in general the public-eye pressure and the importance of the service will prevent the DeLorean provider from modifying any previous volumes without being immediately caught, thus effectively deterring from modifying the history or injecting "fake news" in the past. For example, it would be impossible for DeLorean to modify the record for October 22, 2015 issue of USA Today or deny its existence without actually using a time machine and altering the reality. At the same time, in Section 7.4 we discuss ideas of actions that public auditors can take to inform the public of the inconsistent behavior and what can be done to recover from a failure.

To audit the consistence of the chronicle, auditors need to occasionally retrieve the current chronicle digest (i.e. the hash of the root node) and check whether it "covers" a chronicle digest that the auditor has retrieved before. Overall, to assure correctness of time information, each published volume should be verified by at least one auditor / data producer around the time the volume was published (we discuss the number of auditors needed to achieve this goal in Section 6.3). Once the auditor verifies the consistence between the two digests, he/she can keep the new digest and discard the old one. Therefore, the auditor's storage overhead can be kept constant.

Similar to the existence verification (see Section 5.3), the consistence verification is to re-compute the new root hash from the old one, retrieving missing nodes along the way. With all the nodes of the chronicle tree being published, an auditor can retrieve the nodes that are necessary for consistence verification in at most $O(\log V + \log S)$ number of iterations.

6 ANALYTICAL EVALUATION

To understand the scalability properties of NDN DeLorean, we performed a series of analytical evaluations. In particular, we devised the formulations for (a) the amount of storage capabilities necessary at the DeLorean service provider depending on the timeslot duration of a volume and the amount of fingerprints within a volume, (b) the amount of data an average user needs to retrieve to verify the existence of a data item at a certain point in time, and (c) the number of auditors and the number of audits per day per auditor needed on average to ensure that each volume in the chronicle is verified for correctness.

It is important to point out two key aspects of our evaluation (or rather NDN DeLorean): First, we assume that the system will not be used for *every* data item in an actual NDN deployment, but rather only for *certain* data items for which proof of existence is of *crucial value.* For instance, music downloaded by a user (from a service such as iTunes) normally would not require proof of existence. However, a newspaper article (in particular in the age of *fake news*) may benefit a lot from the NDN DeLorean service, as it would be possible to verify in the future that a certain authority vouched for the correctness of the article at a certain point in time.

Second, as pointed out earlier, even for a very large volume of data represented as a big set of NDN data packets, a public signature of a single manifest file [3] can provide strong authenticity and provenance properties for the whole dataset. Therefore, creating a DeLorean proof-of-existence just for the manifest packet alone would be sufficient to ensure that the whole data volume can be authenticated at any point in the future. Note that this does necessarily mean that archived data must be in the format of manifest. Producers can create a manifest separately and log the signature of manifest for proof-of-existence.

Moreover, to justify the practicability of DeLorean for data archives, we reference the statistics of a real-world archive system: newspaper archive in public libraries. According to Statista, there are 1,331 daily newspapers in the United States in 2014 [21]. A rough estimation shows that every day a mainstream newspaper (e.g., Washington Post, New York Times, etc.) publishes 200 to 500 pieces of contents [16], including stories, graphics, blog posts, etc. Based on these numbers, we estimate at most 700,000 pieces of newspaper content published per day, and on average 486 pieces of content per minute. In contrast, according to American Library Association, there are 9802 public libraries in the United States in 2012 [1], which can serve as auditors for newspaper content.

6.1 DeLorean Service Storage Requirements

Here we estimate the overall storage requirements for the NDN DeLorean service (Γ), for the chronicle-tree (γ_c), and for a volume-tree (γ_v), with notation explained in Table 1. Time t is the age of the chronicle. Note that the leaves of the chronicle and volume trees are not represented as NDN data packets: their hashes are simply included in data packets representing the first-level tree nodes. Therefore, equations 3 and 4 only account for intermediate nodes of trees. In our analysis, we assume $k = m = 32$, as suggested in Section 5.1.

Table 1: Symbols used in Evaluation

Symbol	Explanation
$\Gamma(t)$	Overall storage requirement at time t
$V(t)$	The number of the leaf nodes (volumes) in the chronicle tree at time t
$\gamma_c(t)$	Storage requirement for the chronicle tree at time t
k	The arity of the chronicle tree
$\lvert S\rvert$	The average number of the leaf nodes (signatures) in a volume tree
$\gamma_v(S_i)$	Storage requirement for a volume tree i
m	The arity of the volume trees
σ	Size of node data packet (in bytes)
Δ	The duration of a timeslot within a volume

$$\Gamma(t) = \gamma_c(t) + V(t) \times \gamma_v(|S|) \tag{1}$$

$$V(t) = \lceil t/\Delta \rceil \tag{2}$$

$$\begin{aligned}\gamma_c(t) &= \sigma \times \left(\left\lceil \frac{V(t)}{k} \right\rceil + \left\lceil \frac{V(t)}{k^2} \right\rceil + \ldots + \left\lceil \frac{V(t)}{k^{\lceil \log_k V(t) \rceil}} \right\rceil \right) \\ &= \sigma \times \sum_{j=1}^{\lceil \log_k V(t) \rceil} \left\lceil \frac{V(t)}{k^j} \right\rceil\end{aligned} \tag{3}$$

$$\gamma_v(|S|) = \sigma \times \sum_{j=1}^{\lceil \log_m |S| \rceil} \left\lceil \frac{|S|}{m^j} \right\rceil \tag{4}$$

Equations 1-4 give quasi-linear storage requirement growth over time. To highlight the yearly requirements, we calculated a 20-year storage requirement for different witnessed signature volumes and averaged the resulting value over the years (year-to-year variation was less than 0.01%). Figure 8 shows the result of timeslot size Δ = 10 minutes.

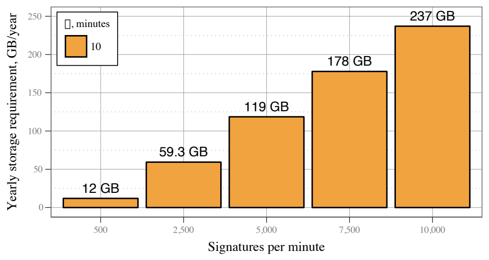

Figure 8: NDN DeLorean service storage requirements

For a timeslot duration of $\Delta = 10$ minutes, one year of DeLorean service requires creation of about 52,560 volumes (equation 2), which can be summarized using about 1698 Chronicle tree nodes (equation 3). This number will be slightly smaller or larger depending on how many new intermediate nodes need to be created and whether the tree grows a level. Overall, each year the Chronicle tree (only complete nodes and the latest versions of incomplete nodes) would require $\approx$ 2.5 MB or fewer if a longer timeslot is used. At the same time, depending on how many signatures DeLorean service records per timeslot, the storage requirements can significantly increase. For example, when recording 500 signatures per minute (according to the statistics of newspaper content [16] discussed in the introduction of Section 6, and assuming it is constant over the years, equations 1 and 4), 12 GB storage per year for the volume trees will be needed and this value will dominate the storage cost.

When a larger timeslot is used, a slightly smaller amount of space is needed. For example, the storage requirement for 60-minute Δ is 11.9 GB per year for 500 signatures per minute and just under the value for $\Delta = 10$ minutes for higher signature rates. This difference is mainly because storing a smaller number of signature fingerprints in the volume is less space-efficient: more incomplete nodes resulting in higher overhead for NDN data packet name and signature.

Note that the yearly storage requirement is linear to the number of witnessed signatures, i.e., it depends on the popularity of the DeLorean service and can grow or shrink over time.

6.2 Verification Cost

The amount of data a user would need to retrieve to verify the existence of a signature at a certain point in the past, r_u, is expressed in equations 5-7. A user needs to retrieve certain parts of the chronicle-tree (r_c) and certain parts of the actual volume (r_v) to which the signature in question belongs, according to the time it was published. Since we store the hash values of a node's children in a node's NDN data packet, for each level in the tree one node needs to be retrieved.

$$r_u = r_c + r_v \tag{5}$$

$$r_c = \sigma \times \lceil \log_k V(t) \rceil \tag{6}$$

$$r_v = \sigma \times \lceil \log_m |S| \rceil \tag{7}$$

For $k = 32$, the height of the Chronicle tree would change very infrequently, defining almost a constant overhead for the r_c. For a 10-minute timeslot, this would be just three 1500-byte data packet retrievals for the first 20 years of the DeLorean service and four retrievals for foreseeable future (> 600 years). r_v depends on the number of signatures per volume, but is also logarithmically scaled: for 500 signatures per minute, it would account for another three retrievals.

The same considerations apply for storage costs for a verifier. In order to ensure DeLorean chronicle consistency, the retrieved data packets need to be stored until the next verification, requiring a trivial overhead, generally fewer than 20 KB.

6.3 Required Number of Auditors

The trustworthiness of the DeLorean service comes from the constant verification of the published chronicle by a set of decentralized auditors. In this section, we assess how many auditors are needed in the system to provide reasonable service assurances. In a distributed system with volunteering auditors, there is no guarantee that *all* auditors behave reliably all the time. Thus, the basis of our evaluation is the probability that there is a volume that has not been verified by at least one auditor around the time the volume has been finalized. The existence of such a timeslot means that the DeLorean service could have published the volume at a different time than claimed.

We assume that each of the auditors in a set A fetches and verifies the chronicle at least once during an epoch e. Each fetch happens at a random point of time within the epoch, stochastically independent from all the other auditors. We consider the following event probabilities:

- $P(NV)$: "Probability that a volume in the epoch has not been verified by any of the |A| auditors" (equation 8),
- $P(V_1)$: "Probability that a volume in the epoch has been verified by at least one of the |A| auditors" (equation 9), and
- $P(V_A)$: "Probability that all volumes in a given epoch have been verified by at least one auditor" (equation 10).

$$P(NV) = \left(1 - \frac{1}{e/\Delta}\right)^{|A|} = \left(1 - \frac{\Delta}{e}\right)^{|A|} \tag{8}$$

$$P(V_1) = 1 - P(NV) \tag{9}$$

$$P(V_A) = P(V_1)^{e/\Delta} \tag{10}$$

Figure 9 visualizes the probability results for V_A versus the numbers of auditors $|A|$ for different epoch period lengths e and timeslot durations Δ. It is clear that when there are not enough verification events, the trustworthiness of the DeLorean service can be really low, as in the case of 100 auditors performing only a single verification a day of the Chronicle with 10-minutes timeslots. At the same time, the same number of auditors could be sufficient to provide $\approx$ 75% chance that all 60-minute volumes within a day are checked. Of course, the larger population of auditors ensures higher probabilities of consistency verification and, therefore, ensures better trustworthiness of the DeLorean service. For example, about 2000 auditors (much less than 9081 auditors in the newspaper archive example) would be enough to reliably verify chronicle with 1-hour time intervals, even when each of the auditors performs only one verification a week.

7 DISCUSSION

7.1 Multiple DeLorean Providers

The example presented in this paper only involves a single instance of DeLorean. In deployment multiple instances of the DeLorean service would run in parallel, with data publishers making

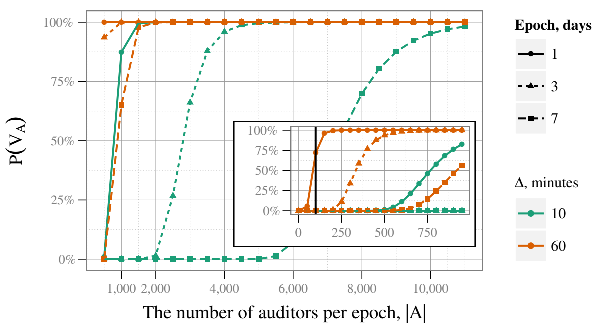

Figure 9: DeLorean audit probabilities

individual decisions on which specific instance to use. We envision the DeLorean service to be provided by service providers either as a voluntary community service (e.g., similar to Certificate Transparency [10]), or as a fee-based service (e.g., based on micropayments). The only change to the described protocol would be in the naming: instead of a single "`/DeLorean`" prefix, the chronicle volumes will be published under "`/google/DeLorean`", "`/apple/DeLorean`", and similar prefixes. However, note that the consistency guarantee of a single DeLorean instance depends on the quality of the public audit, discussed in Section 6.3. We envision that in addition to the public-eye audit, the DeLorean instance will be cross-verified by major service providers, as they have incentives to catch a competitor's wrongdoings and to avoid being caught by others.

7.2 Incentives to Audit

Users of the system have a clear interest in its trustworthiness: a misbehaving service provider or one with a bad reputation is of low value to them (in convincing consumers of the validity of their signature in the future). Therefore, every user should have a natural interest in becoming an auditor. In reality, however, it is likely that only a fraction of users actually will act as auditors. Estimating the precise fraction of users being willing to audit is challenging. Our calculations on the number of necessary auditors in Section 6.3 show that a comparatively small number of auditors suffices to assess the trustworthiness of the service provider.

7.3 Chronicle Resiliency

In addition to traditional methods of providing service resiliency using DeLorean instance replication, DeLorean leverages the power of NDN to disseminate and preserve the data. In other words, all DeLorean data is published as NDN data packets and is constantly retrieved by the publishers (to retrieve just issued signature proofs), the auditors (to verify consistency), and consumers (to verify proofs). Even if the primary servers of the DeLorean provider become (temporary) unresponsive, the already issued proofs can be retrieved directly from the in-network caches and/or from producers', auditors', and consumers' records. Given the modest storage requirement, we expect that in addition to cross-verification in the multi-provider deployments, the providers will simply mirror the complete chronicle data of each other, improving the overall resiliency to potential failures even more.

7.4 Recovery from Inconsistency

Whenever an auditor detects that the DeLorean provider is not consistent with the previously recorded states, it needs to take actions. These actions include contacting the provider to remediate the issue, as well as notify publishers, consumers, and other auditors of the service about the detected problem. This course of actions is another side of the security-through-publicity concept [18], requiring an open public channel between auditors, system users, and system providers, e.g., a simple *bulletin board* or a blog.[6] Through this channel, the problem can be confirmed by multiple auditors, forcing the provider to immediately address the problem or risk to lose its business. Note that auditors cannot falsely accuse a DeLorean service of failure or intentional modification of the chronicle. Because every data packet published by the DeLorean service bares the service's signature on the root node data packet, an auditor can claim the inconsistency only by presenting two (or more) data packets signed by the service that show the inconsistent state.

In an unlikely case when the DeLorean provider does not respond to the reported issues or no longer wishes to provide the service, it is possible to transition to a new provider. Recall that the authenticity of previous volumes can be implicitly verified through Merkle tree state (i.e., root digest). The new provider can pick up the service from a state under the consensus of the auditors and obtain copies of the existing volumes that represent the last consistent state. After updating the publisher public key, users of the timestamp service can keep using the same historical volumes, and start re-creating volumes.

7.5 Preventing Multiple Histories

As mentioned before, one of the design objectives of DeLorean is to reinforce the trust of the service by auditors' consistency verification. This ensures that chronicle publisher cannot modify the history without being detected by auditors. However, if chronicle publisher could identify the source of audit requests, the publisher can cheat by presenting one consistent chronicle to a group of auditors while presenting a completely different but still consistent chronicle to another, which is usually called a *multiple history issue*. In this case, none of the auditors in the two groups can detect any modification, while the timestamp service would be obviously inconsistent.

NDN intrinsically eliminates the possibility of identifying an auditor or a group of auditors from the audit request. First of all, the NDN interests for DeLorean data do not carry any information about auditors. Second, interests may be merged by routers (when multiple auditors request state at the same time) or get served by in-network caches, so that DeLorean may not receive the interests from every auditor. For example, an auditor in one group can detect inconsistent history when picking a piece of data that is prepared for another group. Thus, NDN provides some basic protection. We consider further investigation of potential threats in this area and the design of countermeasures (e.g. concrete gossiping solutions) as interesting future work.

[6]Potential Denial-of-Service (DoS) attacks on such a bulletin board with an overwhelming amount of false claims by auditors are out of scope of this paper.

7.6 Relation to Real-Time Data Production

Note that creating proofs-of-existence for signatures is an additional procedure to ensure data can still be authenticated after the signature expires. Therefore, it has not effect on real-time data production and consumption: before the original signature of a data packet expires, consumers can directly verify the data without needing the timestamp. For our example, "Youth Jailed" article of USA Today could have been directly authenticated on October 22, 2015 or several days after, while the original signature is still valid. Only when the readers access that article year or so later, they may need to use the timestamp proof in order to ensure the authenticity of the article at the time it was originally published.

7.7 Hash Rollover

For the sake of simplicity, we used only one hash algorithm (SHA-256) in the description above. However, no hash algorithm can be secure forever. To address this issue, a DeLorean service can publish two versions of chronicle tree, of which each is constructed using hash algorithms with different crypto strength. Since it is very unlikely that the two hash algorithms will be broken at the same time, the stronger hash algorithm offers the publisher enough time to find another hash algorithm with more crypto strength and to rebuild the chronicle trees according to the surviving tree. Given hash algorithm breaking happens rarely, the overhead of building and verifying a new tree though can relatively expensive but is still affordable.

8 RELATED WORK

To the best of our knowledge, Haber and Stornetta [8] were the first to propose the use of the timestamp service to secure digital documents. They built the service by linking documents in a time order using a crypto hash function, allowing users to check the existence of a document by checking against a set of documents along with the timeline. Buldasi et al. [4] later proposed a binary linking timestamp that simplified implementation of the timestamp service. Additional information and history of the timestamp service designs are available in the survey by Vigil et al. [22]. RFC4810 lists (among others) requirements for such timestamping services [23].

The timestamp service work that is most related to DeLorean design is KASTS [12]. KASTS not only timestamps signed documents, but also keeps a secure storage of verification keys. Compared to KASTS that builds the timestamp service over hash chains, DeLorean uses Merkle tree hierarchy to allow efficient public auditing. Moreover, KASTS is focused on a single trust model, i.e., the PKI model, while DeLorean supports data signing under arbitrary trust models, as long as consumers know the corresponding trust schema.

The foundation of DeLorean design is a work of Crosby and Wallach [6] that proposes the use of Merkle tree to implement a tamper-evident logging system. They conducted the detailed performance analysis to prove the efficiency of the Merkle tree based logging system. They also proposed a scheme to safely delete log entries that are no longer needed, which we plan to investigate in the next revisions of the DeLorean design.

Certificate Transparency (CT) [5, 10] offers one of the most important use cases of Merkle tree based logging system and inspired the design of DeLorean. CT is designed to mitigate the certificate mis-issuance problem through a "security through publicity" approach. CT uses Merkle tree to build a public board, on which certificate authorities are required to post all the issued certificates. Using this board, the legitimate owners of the domain names can easily detect the mis-issued certificates. DeLorean borrows the same "security through publicity" concept, but applies it to verification of absolute time of the chronicle volumes: any attempt to "back-publish" or modify volume will be detected by a set of public auditors. Because of the nature of IP protocol, CT instance will always know the source address of the requester. To avoid the problem of multiple consistent views to different users, CT design includes an additional gossip protocol [5]. DeLorean intrinsically avoids this problem by being an NDN-based system: data retrieval in NDN does rely on source addresses, but uses states set up by the incoming requests.

BitCoin [17] represents another example of "security through publicity" approach to support a consistent append-only log based on hash chain. However, BitCoin requires an efficient peer-to-peer overlay multicast network and also requires each peer in the system to keep a copy of the history, thus making it unsuitable for maintenance of a large amount of long-lived data.

9 CONCLUSION

In this paper, we presented the design of NDN DeLorean, the authentication framework for the long-lived data archives in NDN. To our best knowledge, it is the first design to address the long-lived data authenticity challenges in ICN. DeLorean enables secure authentication of the long-lived data considering the times when the original signatures of the data were valid. At the heart of DeLorean is a publicly auditable bookkeeping service that keeps a record of data signatures and the times these signatures were witnessed by the service. More specifically, the collected signatures (their fingerprints) during the pre-defined time intervals are aggregated into the DeLorean volumes, and each volume is getting attached to the data chronicle, both organized as multi-ary Merkle trees. This structure ensures that once an item is added (signature fingerprint into the volume, volume hash into the chronicle), it cannot be removed or replaced with another without changing the overall state of the DeLorean chronicle. Trustworthiness of the DeLorean service is based on the public audit of the record consistency over time, which can be accomplished in time- and space-efficient manner, as verified by a set of analytical evaluations for modest rates of witnessed signatures.

DeLorean is our first step toward securing long-lived data in NDN and lays the foundation for our future of trust management in NDN. As the next step, we plan to continue our investigation, as well as to seek help from the research community to complete the construction of a generic authentication system for all kinds of long-lived data, including public and confidential datasets, and investigate how to mitigate keys' compromise and revocation during their validity periods.

ACKNOWLEDGMENTS

This work is partially supported by the National Science Foundation under award CNS-1345318 and CNS-1629922.

REFERENCES

[1] American Library Association. 2015. Number of Libraries in the United States. http://www.ala.org/tools/libfactsheets/alalibraryfactsheet01, Last accessed: May 6, 2017. (2015).

[2] Richard Barnes, Jacob Hoffman-Andrews, and James Kasten. 2015. Automatic Certificate Management Environment (ACME). https://tools.ietf.org/html/draft-ietf-acme-acme-02. (October 2015).

[3] Mark Baugher, Bruce Davie, Ashok Narayanan, and Dave Oran. 2012. Self-verifying names for read-only named data. In *Proceedings of IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 274–279.

[4] Ahto Buldasi, Peeter Laud, Helger Lipmaai, and Jan Villemson. 1998. Timestamping with Binary Linking Schemes. In *CRYPTO'98*.

[5] Laurent Chuat, Pawel Szalachowski, Adrian Perrig, Ben Laurie, and Eran Messeri. 2015. Efficient gossip protocols for verifying the consistency of certificate logs. In *Communications and Network Security (CNS), 2015 IEEE Conference on*. IEEE, 415–423.

[6] Scott A Crosby and Dan S Wallach. 2009. Efficient Data Structures For Tamper-Evident Logging. In *Security Symposium*. Usenix, 317–334.

[7] Tim Dierks and Eric Rescorla. 2008. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246. (August 2008).

[8] Stuart Haber and W Stornetta. 1990. How to Time-Stamp a Digital Document. In *CRYPTO'90*.

[9] Stephen Kent and Karen Seo. 2005. Security Architecture for the Internet Protocol. RFC 4301. (December 2005).

[10] Ben Laurie. 2014. Certificate transparency. *Queue* (2014).

[11] Ben Laurie, Adam Langley, and Emilia Kasper. 2013. Certificate Transparency. RFC 6962. (June 2013).

[12] Petros Maniatis and Mary Baker. 2002. Enabling the Archival Storage of Signed Documents. In *the USENIX Conference on File and Storage Technologies (FAST) 2002*. Usenix.

[13] Manuel Araoz. 2017. Proof of Existence. https://proofofexistence.com, Last accessed: Aug. 20, 2017. (2017).

[14] Spyridon Mastorakis, Alexander Afanasyev, Yingdi Yu, Michael Sweatt, and Lixia Zhang. 2017. nTorrent: BitTorrent in Named Data Networking. In *ICCCN*.

[15] Ralph C Merkle. 1980. Protocols for Public Key Cryptosystems. In *Security and Privacy, 1980 IEEE Symposium on*. IEEE, 122–122.

[16] Robinson Meyer. 2016. How Many Stories Do Newspapers Publish Per Day? https://www.theatlantic.com/technology/archive/2016/05/how-many-stories-do-newspapers-publish-per-day/483845/, Last accessed: May 6, 2017. (2016).

[17] Satoshi Nakamoto. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. (2008).

[18] Eric Osterweil, Daniel Massey, Batsukh Tsendjav, Beichuan Zhang, and Lixia Zhang. 2006. Security Through Publicity. In *HotSec*.

[19] Zach Shelby, Klaus Hartke, and Carsten Bormann. 2014. The constrained application protocol (CoAP). RFC 7252. (June 2014).

[20] Diana Smetters and Van Jacobson. 2009. *Securing network content*. Technical Report. Citeseer.

[21] Statista Inc. 2017. Number of daily newspapers in the United States from 1985 to 2014. https://www.statista.com/statistics/183408/number-of-us-daily-newspapers-since-1975/, Last accessed: May 6. (2017).

[22] MartÃŋn Vigil, Johannes Buchmann, Daniel Cabarcas, Christian Weinert, and Alexander Wiesmaier. 2015. Integrity, authenticity, non-repudiation, and proof of existence for long-term archiving: a survey. *Elsevier Computers & Security* (2015).

[23] Carl Wallace, Ulrich Pordesch, and Ralf Brandner. 2007. Long-Term Archive Service Requirements. RFC 4810. (March 2007).

[24] Yingdi Yu, Alexander Afanasyev, David Clark, kc claffy, Van Jacobson, and Lixia Zhang. 2015. Schematizing Trust in Named Data Networking. In *Proceedings of the 2nd International Conference on Information-Centric Networking*. ACM.

[25] Yingdi Yu, Patrick Guo, and Alexander Afanasyev. 2017. Source code of NDN DeLorean. https://github.com/named-data/ndn-delorean. (May 2017).